Facilitating Deep Learning for Edge Computing: A Case Study on Data Classification

Abdullah Alsalemi
Institute of Artificial Intelligence
De Montfort University
Leicester, United Kingdom
P2621877@my365.dmu.ac.uk

Abbes Amira
College of Computing and Informatics
University of Sharjah
Sharjah, United Arab Emirates
aamira@sharjah.ac.ae

Hossein Malekmohamadi
Institute of Artificial Intelligence
De Montfort University
Leicester, United Kingdom
h.malekmohamadi@dmu.ac.uk

Kegong Diao
Institute Of Energy And Sustainable Development
De Montfort University
Leicester, United Kingdom
kegong.diao@dmu.ac.uk

***Abstract*—Deep Learning (DL) is empowering technology in a plethora of ways, especially when big data processing is a core requirements. Many challenges, however, arise when solely depending on cloud computing for Artificial Intelligence (AI), such as data privacy, communication latency, and power consumption. Despite increasing popularity of edge computing, the overarching issue in this scope is lack of a comprehensive documentation on how to setup a given edge computing device to run DL algorithms. Due to its specialized nature, installing the full version of TensorFlow DL library on an edge device is a complicated procedure that is seldom successful, due to the many dependent software libraries needed to be compatible with varying architectures of edge computing devices. Henceforth, in this paper, we present a novel guide on how to setup the TensorFlow Lite software library and outline a complete workflow of model training, validation, and testing on the ODROID-XU4. Results are presented for a case study on energy data classification using the outlined model show almost 7 times higher computational performance compared to cloud-based ML.**

Keywords—Edge computing, artificial intelligence, deep learning, data classification, computational offloading.

I. Introduction

Deep Learning (DL) is empowering technology in a plethora of ways, especially when big data analytics is a core process [1]. For instance, cloud computing servers are considered as powerful computational platforms that can perform cost-efficient and scalable Machine Learning (ML) and DL algorithms [2], [3]. Many challenges, however, arise when solely depending on cloud computing for Artificial Intelligence (AI), such as data privacy, communication latency, and power consumption [4]–[6]. Therefore, Edge AI has emerged as a solution in which ML algorithms are run at high performance by securely leveraging local computational resources.

As an example, low power consumption and compact form factor are features of which the Single-Board Computing (SBC) platform ODROID-XU4 is popularly known for. It supports a composite of ARM Cortex-A15, Cortex-A7 big.LITTLE Central Processing Unit (CPU), and ARM Mali-T628 GPU, in addition to several open-source operating systems, such as Linux and Android versions [7].

On the other hand, a well-known AI accelerator, called Jetson Nano, was developed by Nvidia supporting Graphics Processing Unit (GPU) cores of 128 Maxwell. It is used in a variety of edge AI applications with a 5 to 10 watts electric board consumption. For instance, 4 ARM cores and 256 CUDA Maxwell cores are supported by the edge AI accelerator NVIDIA Jetson-TX1, whereas 6 ARM cores and 256 CUDA Pascal cores are supported by the edge AI accelerator Jetson-TX2 [8].

Despite increasing popularity, the overarching issue in this scope is lack of a comprehensive guide on how to setup a given edge computing device to run DL algorithms. Due to its specialized nature, installing the full version of TensorFlow deep learning library on an edge device is a complicated procedure that is seldom successful, due to the many dependent software libraries needed to be compatible with varying architectures of edge computing devices.

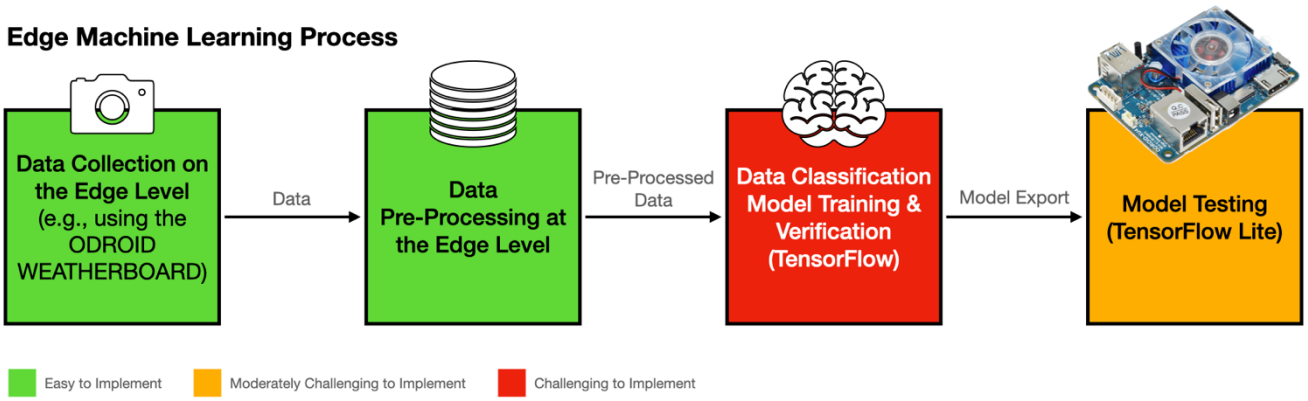

Fig. 1. Edge ML Data Workflow and Challenge Levels.

Henceforth, in this practice paper, a novel guide on how to setup the TensorFlow Lite software library and outline a complete workflow of model training, validation, and testing on the ODROID-XU4. This is the first updated guide for the 2021-2022 software versions of the ODROID-XU4, TensorFlow Lite and accompanying libraries to outline specific steps for enabling DL application using Python. Potential applications include the following:

1. Energy consumption data analysis;
2. Image recognition;
3. Natural language processing;
4. Transfer learning applications; and
5. Condition monitoring and fault detection, among others.

The remainder of this paper is outlined as follows: Section II explores recent related work to edge computational offloading. Section III describes the methods of implementing edge ML on the ODROID-XU4. Section IV showcase a case study on using edge ML on a data classification problem. Section V concludes the paper.

II. Related Work

In this section, we will outline recent research on the area of offloading computation to the edge and its importance on reducing the burden on traditional computational units such as ML cloud computing servers.

According to [9], despite its importance, there is no systemic, exhaustive, or thorough survey of computation offloading processes in the MEC context. In the review in [9], a study of ML-based computation offloading processes is presented in the MEC setting in the context of a classical taxonomy in order to define contemporary frameworks on this critical subject and to raise open concerns. The suggested taxonomy is divided into three major categories: mechanisms focused on reinforcement learning, regulated learning mechanisms, and unsupervised learning models. Following that, these groups are contrasted to one another depending on key features, such as success measurements, case studies, methods used, and assessment tools, as well as their advantages and disadvantages.

In that direction, a DL system for IoT edge devices with a novel offloading scheme is presented in order to optimize ML performance for the resource-constrained nature of edge nodes. The performance of running DL programs on an edge device are tested with experimental results indicate that the suggested methodology improves upon alternative optimization models.

A reoccurring theme in edge AI is the problem of constrained computational power, as well as limited storage capacities compared to cloud alternatives. This is achieved by using the MEC network and using the vast data scattered through a wide range of edge computers. Modeling upon distributed data and communication between the edge nodes are two crucial elements of such systems. The authors call for the needs of novel architectures for wireless connectivity in edge AI. Examples are given to illustrate the feasibility of the suggested criteria of such new architectures, and specific testing metrics are described.

On a similar note, despite the strong contributions found in the field of edge computing, the contributions related to how to actually implement the algorithm are left unintentionally obfuscated and vague, especially when considering boards such as the ODROID-XU4, which creates a gap and an opportunity for prospective research efforts and guidance.

III. Methods

A. Overview and Model Training

The overall data workflow is illustrated in Fig. 1. Assuming the first two steps are fulfilled, i.e., data is collected and pre-processing as per the desired application, we can proceed to model training and validation. Also, the figure highlights different challenge levels at each stage of the process. In this paper, we will propose a method for the last step of the workflow, i.e., executing pre-trained machine learning models on an edge computing device using TensorFlow Lite on ODROID-XU4.

In order to train a DL model, a suitable DL software library must be chosen and installed. In this case study, TensorFlow is employed using Python on the publicly available Google Colab platform[1]. In this implementation TensorFlow V2.7.0 is installed. It is important to note that the mentioned software packages are required based on the prerequisites of the commonly used

Due to the specific scope of this paper, model training and validation are not covered. Depending on the chosen application, we assume a suitable DL model is implemented, training and validated on existing data.

B. Exporting TensorFlow Models to TensorFlow Lite

In order to run the model on an ODROID-XU4 edge computing device, a lightweight version of the TensorFlow model needs to be exported as shown in the Appendix.

This will result in a generating a .tflite file, which is an edge optimized version of the original model.

C. Install TensorFlow Lite on the ODROID-XU4

After a TensorFlow Lite model is exported, the required libraries are installed on the ODROID-XU4 as follows. Given an ODROID-XU4 is setup to have the latest Ubuntu Mate operating system (V21.04 or later) as well as Python V3.8, you can install the needed libraires using the Terminal program as shown in the Appendix.

Following, the TensorFlow Lite model can be run using a specialized Python program that computes the model on a testing sub-dataset. An example created by the authors[2].

Using the workflow described, any compatible DL model can be run on edge computing devices, the ODROID-XU4 in this case study, enabling numerous advantages such as higher performance, lower power consumption, higher data privacy, and minimal application complexity using lightweight models.

IV. Results: case study on energy data classification

In this section, to illustrate the outcomes of this work, an implementation of edge-based ML classifier of energy consumption data is described on the UK Domestic Appliance-Level Electricity (UK-DALE) dataset is employed

[1] https://colab.research.google.com

[2] https://github.com/Abdol/edge-test

due to its large size and multi-year duration [10], [11]. The UK-DALE dataset keeps record of the power consumption data from five houses in the UK, where in each household, aggregated power consumption is collected along with appliance-level data every six seconds. In our implementation, we have used aggregated power consumption data from one house.

Following training and validation on the Google Colab, the model is exported, downloaded, imported, and tested on the ODROID-XU4 following the method described in Section III. It also noteworthy to mention that the TensorFlow Lite model is tested on the cloud for accuracy and computation performance benchmarking purposes. Table 1 compares the model testing performance between the edge ODROID-XU4 running the TensorFlow Lite model, Google Colab running the TF model, and the TensorFlow Lite cloud implementation. In terms of computational speed, the cloud implementation excels given its high performance GPU. However, the performance of the ODROID-XU4 is considerably high with 28.5 sec, which is approximately 6.9 times faster the cloud TensorFlow Lite implementation. In terms of accuracy, the model scores evenly with an average of 89.4%. To provide more perspective, the ODROID-XU4 can classify a GAF image that represents around 42 thousand data points in less than 17.5 msec, which is very close to real-time performance.

Table 1. Data classification performance comparison between edge and cloud implementations.

	ODROID-XU4 (Edge, TFLite)	**Cloud (Google Colab, TF)**	**Cloud (Google Colab, TFLite)**
Model performance on test dataset (sec)	28.51	19.10	196.62
Model accuracy	89.57%	89.65%	89.08%

V. Conclusions

In this paper, the methods of implementing edge-level ML has been described on a case study board with exemplary computational performance that signifies the benefits of computational offloading for data analytics applications. With emphasis on the ODROID-XU4 board for computation offloading purposes, this work tackles one of the most common challenges that research face when working on edge ML, preparing the required software on the desired edge platforms. As the results of energy data classification case study is depicted, the presented workflow is validated and has shown that specific steps can be taken to enable running complex algorithms on the edge computing devices, opening new doors toward broader data analytics applications that are both energy efficient and privacy preserving.

References

[1] C. Fan, Y. Sun, Y. Zhao, M. Song, and J. Wang, "Deep learning-based feature engineering methods for improved building energy prediction," *Applied Energy*, vol. 240, pp. 35–45, 2019.

[2] M. S. Badar, S. Shamsi, M. M. U. Haque, and A. S. Aldalbahi, "Applications of AI and ML in IoT," in *Integration of WSNs into Internet of Things*, CRC Press, 2021.

[3] M. Antonini, T. H. Vu, C. Min, A. Montanari, A. Mathur, and F. Kawsar, "Resource Characterisation of Personal-Scale Sensing Models on Edge Accelerators," in *Proceedings of the First International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*, New York, NY, USA, Nov. 2019, pp. 49–55. doi: 10.1145/3363347.3363363.

[4] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Generation Computer Systems*, vol. 115, pp. 619–640, Feb. 2021, doi: 10.1016/j.future.2020.10.007.

[5] H. Li, J. Yu, H. Zhang, M. Yang, and H. Wang, "Privacy-Preserving and Distributed Algorithms for Modular Exponentiation in IoT With Edge Computing Assistance," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8769–8779, Sep. 2020, doi: 10.1109/JIOT.2020.2995677.

[6] J. Chi *et al.*, "Privacy Partition: A Privacy-Preserving Framework for Deep Neural Networks in Edge Networks," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, Oct. 2018, pp. 378–380. doi: 10.1109/SEC.2018.00049.

[7] M. Merenda, C. Porcaro, and D. Iero, "Edge Machine Learning for AI-Enabled IoT Devices: A Review," *Sensors*, vol. 20, no. 9, Art. no. 9, Jan. 2020, doi: 10.3390/s20092533.

[8] S. M. Sánchez *et al.*, "Edge Computing Driven Smart Personal Protective System Deployed on NVIDIA Jetson and Integrated with ROS," in *Highlights in Practical Applications of Agents, Multi-Agent Systems, and Trust-worthiness. The PAAMS Collection*, Cham, 2020, pp. 385–393. doi: 10.1007/978-3-030-51999-5_32.

[9] A. Shakarami, M. Ghobaei-Arani, and A. Shahidinejad, "A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective," *Computer Networks*, vol. 182, p. 107496, Dec. 2020, doi: 10.1016/j.comnet.2020.107496.

[10] J. Kelly and W. Knottenbelt, "The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes," *Scientific Data*, vol. 2, p. 150007, Mar. 2015, doi: 10.1038/sdata.2015.7.

[11] A. Alsalemi, A. Amira, H. Malekmohamadi, K. Diao, and F. Bensaali, *Elevating Energy Data Analysis with M2GAF: Micro-Moment Driven Gramian Angular Field Visualizations*. International Conference on Applied Energy, 2021. Accessed: Dec. 27, 2021. [Online]. Available: https://dora.dmu.ac.uk/handle/2086/21303

APPENDIX: ODROID-XU4 TENSORFLOW INSTALLATION COMMANDS

```
model.export(export_dir='model.tflite')
sudo apt-get update

sudo apt-get install python-pip python-numpy swig python-dev

sudo pip3 install tflite-runtime cython h5py numpy pandas

sudo pip3 install numpy

sudo python3 your-tflite-script.py
```